Grado en Ingeniería Informática

Trabajo Final de Grado

Desarrollo de e-commerce con integración de plataforma formativa y redes sociales

Autor:
Miguel Ángel Cebrián Marzo

Supervisor:
Jorge José Bono Bru
Tutor académico:
M. Ángeles López Malo

Fecha de lectura: 9 de noviembre de 2016
Curso académico 2015/2016

Resumen

Este documento presenta el proyecto desarrollado para la empresa Formación Lanzanet que pretende utilizar dos plataformas web conocidas de software OpenSource, Moodle y Prestashop, para unificarlas y crear una plataforma web donde poder presentar y vender los cursos de que dispone la empresa.

Se pretende utilizar la plataforma de Prestashop para presentar los distintos cursos y utilizar las herramientas que posee para gestionar las ventas de los cursos. La plataforma Moodle facilita que se muestre toda la información educativa que posee la empresa y conseguir unificar ciertos aspectos de ambas plataformas para que estas se utilicen en conjunto.

Palabras clave

Plataforma web, Prestashop, Moodle, módulos, e-commerce.

Keywords

Web platform, Prestashop, Moodle, plugins, e-commerce.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

En este documento se presenta el proyecto desarrollado para la empresa *Formación Lanzanet* que pretende utilizar dos plataformas web conocidas de software OpenSource, Moodle y Prestashop, para unificarlas y crear una plataforma web donde poder presentar y vender los cursos de que dispone la empresa. *Formación Lanzanet* es una empresa presente en la población de Sagunto desde el año 1997 que ofrece servicios de formación tanto en el ámbito de la informática como en el de los estudios académicos. En el año 2013, amplía sus servicios como empresa consultora informática, desarrolladora de páginas web, posicionamiento en buscadores (SEO), adaptación a LOPD, tiendas online (e-commerce) y de Marketing en las Redes Sociales mediante campañas publicitarias.

Esta empresa posee diferentes cursos de formación que han ido obteniendo a lo largo de los años, y pensaron en una plataforma de gestión de cursos online donde almacenar estos cursos y que les permita a los alumnos acceder fácilmente a su contenido. Este contenido puede ser variado, desde vídeos-tutoriales accesibles desde un servicio externo como YouTube o Vimeo, apuntes en formato PDF y ejercicios que se podrán realizar en la propia plataforma.

También tienen en mente utilizar una tienda online donde se podrán vender estos cursos, mostrando información sobre estos y facilitando su compra. Esta tienda online estará conectada con la plataforma de los cursos y matriculará a los usuarios en los cursos que adquieran.

Para facilitar que los usuarios accedan a esta plataforma, se desea que aparezca en una buena posición en los principales motores de búsqueda, y tenga su presencia en las redes sociales más conocidas.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto es ofrecer un e-commerce donde vender los diferentes cursos online que posee la empresa y una plataforma web donde mostrarlos.

Dentro del objetivo principal se puede desplegar diferentes subobjetivos:

- Diseño, desarrollo y puesta a punto del e-commerce.
- Facilitar que el administrador pueda crear los cursos online que dispone la empresa.
- Sincronizar las cuentas de usuario de la tienda online y la plataforma web.
- Facilitar el pago de los cursos y mostrar los cursos adquiridos en la plataforma web.
- Optimizar el portal web para que tenga un buen posicionamiento en los motores de búsqueda (SEO).
- Conexión con las RRSS para publicitar la plataforma.

1.3. Alcance

Al final de este proyecto se desea tener un portal web donde un alumno pueda matricularse en diferentes cursos utilizando su cuenta de usuario. En el portal se muestra información acerca del curso y su precio si es de pago. En el caso de querer matricularse en un curso de pago, se le facilitara una pasarela de pago virtual donde pueda realizar la compra. También tendrá una plataforma donde podrá ver el material educativo de los cursos en los que se ha matriculado. El administrador del portal web gestionará los datos de los cursos a través una interfaz web.

1.4. Estructura de la memoria

Este documento se divide en varios capítulos en el que en cada uno describirá una parte del desarrollo del proyecto. En el capítulo 2 se describirá el proyecto mostrando las tecnologías usadas y su situación inicial. En el capítulo 3 se mostrara la planificación en la que este proyecto ha sido desarrollado, enseñando su metodología, su planificación y la estimación de los recursos necesarios. En el capítulo 4 se observa el análisis de los componentes del sistema y su interfaz. En el capítulo 5 se explicarán los detalles de la implementación del proyecto. Y por último, en el capítulo 6 las conclusiones a las que se ha llegado al finalizar el proyecto.

Capítulo 2

Descripción del proyecto

Este proyecto se divide en dos partes, la parte de compra online (e-commerce) y la parte donde gestionar los cursos online. El e-commerce servirá para mostrar los cursos disponibles, tanto los gratuitos como los de pago, y facilitar la venta de estos cursos a través de una pasarela de pago. Cada cierto tiempo se enviarán mensajes publicitarios a través de las redes sociales y directamente a las cuentas de los estudiantes interesados en obtener información sobre los cursos, este mensaje publicitario se plantea que sea manual, el *Administrador* seleccionará los cursos que quieran publicitarse.

La plataforma de gestión de cursos online servirá para mostrar el contenido de los cursos. Se espera que dentro los cursos contengan diferentes ficheros referentes al tema del curso, como por ejemplo, videos explicativos, PDFs, enlaces de interés o cuestionarios, entre otros.

Para poder acceder y/o comprar los cursos, se debe crear una cuenta de *Estudiante* y matricularse en los cursos a través de ella. Se modificará el e-commerce y la plataforma donde están ubicados los cursos para que compartan la misma cuenta de usuario, evitando así crearse dos cuentas por separado y los problemas que puedan surgir por ello. Los encargados de administrar y crear los cursos serán el *Administrador* y los *Profesores*. En un principio habrá una única persona que sea el *Administrador*, pero a medida que vaya creciendo el numero de *Alumnos* será necesario tener encargados de los cursos, de ahí la necesidad de crear el rol de *Profesor* para que sean capaces de gestionar más cursos. El *Administrador* será el encargado de crear las cuentas de *Profesores*.

El *Administrador* también se encargará de gestionar el contenido del frontend de ambos sitios a través del backend. Cada parte es independiente por lo que tendrá su propio backend y frontend. Dentro del backend del e-commerce podrá administrar todos los datos referentes a la venta de los cursos (su precio, una pequeña descripción, y si tiene, añadir un pequeño vídeo descriptivo, etc.) y a las compras de los *Alumnos* (los cursos comprados, sus carritos de compra, etc.), mientras en la parte de la gestión de cursos administrará la parte de los datos de los *Profesores* (los cursos creados por ellos), las matriculaciones de los *Alumnos* (los cursos inscritos) y los cursos (su contenido, los alumnos inscritos y el profesor asignado).

Para la realización de este proyecto se ha tenido en cuenta las recomendaciones del supervisor

en utilizar proyectos Open Source que cumplen parte de la funcionalidad deseada. Para la parte referente a la compra online de los cursos se utilizará PrestaShop y para la parte de gestión de los cursos, Moodle.

PrestaShop [1] es una plataforma Open Source que permite la creación de tiendas online. Está escrita en PHP y tiene soporte con base de datos MySQL. Otros lenguajes que utiliza son JavaScript, HTML, CSS, XML y un template llamado Smarty. Utiliza un patrón Modelo-Vista-Controlador (MVC), adicionalmente utiliza tecnologías como jQuery, Boostrap, Sass, etc. Prestashop posee una amplia colección de módulos que añaden o modifican funcionalidades que trae por defecto la plataforma.

Moodle [2] es una plataforma Open Source que proporciona un sistema de gestión de cursos online. Está desarrollado en lenguaje PHP y tiene soporte a varias base de datos. Por defecto posee unos roles ya definidos que se utiliza en este proyecto, como los roles de *Profesores* y *Alumnos*.

Ambas plataformas son independientes y habrá que realizar modificaciones en su código para realizar la sincronización de las funcionalidades deseadas. Por poner un ejemplo, al mismo tiempo que se confirma la compra de un curso en la plataforma PrestaShop, se realizará la matriculación del curso en Moodle.

2.1. Prestashop

Prestashop es una plataforma OpenSource que permite crear un e-commerce con unas funcionalidades básicas y facilita añadir nuevas funcionalidades a través de módulos. Dentro de Prestashop se puede distinguir dos partes, el *frontoffice* y el *backoffice*:

- El **frontoffice** representa la parte en la que los *Estudiantes* navegarán por la tienda online, que corresponde a donde se muestran los cursos, una pequeña descripción de los mismos, un carrito de la compra y una pasarela de pago.
- El **backoffice** representa la parte donde el *Administrador* puede controlar lo que ocurre en el *frontoffice*. Aquí se creará la información de los cursos que los estudiantes verán en el *frontoffice* y podrá configurar el comportamiento de la tienda online. También se realiza en esta parte las instalaciones de los módulos y su configuración.

Se espera que el *estudiante* entre en primer lugar al frontoffice de Prestashop, se identifique como usuario y acceda a Moodle a través de un enlace para ver el contenido de los cursos que está matriculado. Si el alumno desea comprar algún curso más, podrá hacerlo a través de Prestashop, ya que se le muestra una pequeña descripción del contenido de los cursos en los detalles de los productos. Además, desde un principio, Prestashop ya tiene implementado el carrito de la compra y la confirmación de los pedidos siendo posible comprar productos sin realizar cambios en la configuración.

El *Administrador* podrá configurar desde el *backoffice*: los productos, las categorías de los productos, las cuentas de usuarios registradas, los pedidos y realizar cambios en la configuración

del comportamiento de la tienda. Además desde el panel de instalación de módulos se puede instalar fácilmente nuevos módulos que puedan añadir una funcionalidad adicional a la tienda, o bien, cambiar la configuración de los módulos ya instalados. Dentro del *backoffice* también se encuentra un sistema de avisos en la parte superior en el que se informa cuando ocurre algún evento, como por ejemplo un usuario se ha creado una cuenta o se ha confirmado algún pedido.

Las cuentas de *Profesor* también podrán acceder al *backoffice*, pero estará limitado únicamente a administrar los productos. Podrán crear productos (cursos) y editar la información que se muestra de los cursos en el *frontoffice*.

2.2. Moodle

Moodle es una plataforma de distribución libre diseñada para ser una plataforma de gestión de cursos. Tras realizar una instalación en nuestro servidor se crea una plataforma funcional y con una base de datos ya estructurada. También se crea una cuenta de **Administrador** en la que ya están configurados unos permisos que permiten ver, editar y configurar toda la plataforma. Los roles que se crean por defecto y que más se utilizan son:

- El **Estudiante** es considerado el rango con menos privilegios. Es capaz de ver los cursos a los que se ha matriculado y no puede realizar modificaciones en la plataforma.
- El **Profesor** es capaz de crear cursos y matricular a Estudiantes. Pueden modificar la información de los cursos que crea y su contenido.
- El **Administrador** puede acceder al panel de administración, además de las funciones de los anteriores roles, y configurar cada aspecto de la plataforma.

Moodle facilita crear cursos para poder añadir el contenido didáctico relacionado con el curso. También facilita a la hora de administrar las matriculaciones de los *estudiantes* en dichos cursos. Cuando un usuario se identifica como *estudiante* podrá acceder fácilmente a una lista con los cursos en los que está matriculado y acceder a la información contenida dentro de los cursos.

Los *profesores* podrán acceder a un panel de control donde podrán configurar los cursos que tengan permisos, donde modifican la configuración y contenido de los cursos. También pueden administrar la lista de los alumnos matriculados en los cursos.

El *administrador*, además de las acciones descritas anteriormente, puede instalar *plugins* que añada alguna funcionalidad nueva y configurarlos. Estos *plugins* modifican algún aspecto que hay de base para añadir una nueva funcionalidad, por poner un ejemplo, un *plugin* para el contenido de los cursos que añade un formulario tipo test para realizar cuestionarios dentro de los cursos.

Capítulo 3

Planificación del proyecto

3.1. Metodología

Dado la facilidad para poder contactar con el cliente del proyecto, el propio supervisor, se ha aplicado una metodología ágil para el desarrollo de este proyecto. Se aplicará la metodología Scrum [3] para la realización de este proyecto. Cada día se realiza una reunión con el supervisor para conocer las tareas a realizar y comentar el estado del proyecto. Cada inicio de semana se realizará una pequeña reunión para indicar las tareas que hay que desarrollar durante la misma y se apuntarán en una pizarra *KanBan*.

La pizarra *KanBan* es una herramienta que ayuda con la gestión de un proyecto. Consiste en tres columnas *To Do*, *Doing* y *Done* en las que se representa los tres estados de una tarea. Las tareas pendientes de hacerse se encuentra en la columna *To Do*, las tareas que se están desarrollando se encuentran en la columna *Doing* y las tareas ya finalizadas se encuentran en *Done*.

Las ventajas de usar una pizarra *KanBan* es que permite ver el flujo de tareas y tener una visión global del estado del proyecto y de las tareas pendientes. Se utiliza la herramienta online Trello por su facilidad para crear columnas y tareas, además de almacenar los cambios realizados en la nube y ser visible en cualquier momento.

Cada inicio de Sprint comenzará se decide qué tareas se realizan durante la duración del Sprint. Aquí se tendrá en cuenta las tareas pendientes y se añadirán nuevas tareas a la pila del producto. Un vez hecha la definición de Sprint se continuará con la implementación de las tareas de la pila de producto y por último se realizará unas pruebas sobre las tareas que se han desarrollado, comprobando que funcionen correctamente.

3.2. Planificación

En la figura 3.1 se muestra una estructura de descomposición del trabajo donde se puede observar una estimación inicial del coste temporal de cada parte del proyecto.

Nº	Tareas	Tiempo (h)	Dependencias
1	**Desarrollo de la propuesta tecnica**	**36**	
1.1	**Inicio**	**8**	
1.1.1	Definir el contenido del proyecto	3	
1.1.2	Definir las tecnologias utilizadas	4	1.1.1
1.1.3	Definir las posibles mejoras para el proyecto	1	1.1.2
1.2	**Planificar el proyecto**	**28**	
1.2.1	Definir las tareas y estimar fechas	4	1.1
1.2.2	Crear diagrama de Gantt	8	1.1
1.2.3	Documentar la propuesta del proyecto	16	1.2.1, 1.2.2
1.2.4	Subir la propuesta tecnica	0	1.2.3
2	**Desarrollo tecnico del proyecto**	**330**	
2.1	**Desarrollo**	**324**	
2.1.1	Sprint 1	60	2.1
2.1.2	Sprint 2	40	2.1.1
2.1.3	Sprint 3	40	2.1.2
2.1.4	Sprint 4	40	2.1.3
2.1.5	Sprint 5	40	2.1.4
2.1.6	Sprint 6	40	2.1.5
2.1.7	Sprint 7	64	2.1.6
2.2	**Puesta en marcha**	**6**	
2.2.1	Implementación	3	2.1
2.2.2	Formación	3	2.1
2.2.3	Entrega final	0	2.2.1, 2.2.2
3	**Documentacion y presentación del TFG**	**145**	
3.1	Redaccion de informes quincenales	4	
3.2	Redaccion de la memoria tecnica	120	
3.3	Entrega de la memoria tecnica	0	
3.4	Preparacion de la presentacion oral	20	
3.5	Presentacion oral	1	

Figura 3.1: Estructura de descomposición del trabajo.

3.3. Estimación de costes

Para el desarrollo de este software se ha utilizado los siguientes recursos:

- 300 horas de trabajo de un programador junior.
- Un ordenador.

- Un servidor de hosting externo.
- Conexión a Internet.

Se ha tenido en cuenta las siguientes condiciones para orientar el precio:

- El coste mensual de una jornada completa de un programador junior sin experiencia laboral es de 2.200 € brutos. [1]
- El valor de €/h se ha calculado tal que 2200 € / (22 días * 8 horas) = 12,5 €/h.
- Se ha hecho un cálculo aproximado de los costes fijos de la empresa como el consumo electrico, amortización de equipos, alquileres, seguros, impuestos, etc. que se valora en 150 € al mes.
- El precio de una conexión a Internet orientado a empresa es de 56,20 €/mes. Según la empresa de telefonía Movistar a la fecha 08/2016. [2]
- El precio al mes del hosting externo es de 4,95 €/mes.
- No se ha tenido en cuenta el sueldo del supervisor de las prácticas.

Se muestra en el cuadro 3.1 el calculo total del coste del proyecto.

	Unidades	**Precio Unitario**	**Total**
Programador Junior	300 horas	12,5 €/h	3.750 €
Hosting Externo	2 meses	4,95 €/mes	9,9 €
Internet	2 meses	56,20 €/mes	112,4 €
Costes fijos	2 meses	150 €	300 €
		Total	**4.172,3** €

Cuadro 3.1: Estimación del coste del proyecto.

3.4. Seguimiento del proyecto

El proyecto se ha divido en 7 Sprints, se planifica que cada Sprint tenga una duración de una semana excepto el 1º Sprint que se realizará en semana y media. La reunión de inicio de Sprint se realizará cada inicio de semana para planificar las tareas que se realizarán a lo largo de esta:

- Sprint 1. 60 horas. 8 de junio hasta 20 de junio.

[1] `http://www.tusalario.es/main/salario/comparatusalario?job-id=2512010000000/`
[2] `http://www.movistar.es/empresas/`

- Sprint 2. 40 horas. 20 de junio hasta 27 junio.
- Sprint 3. 40 horas. 27 junio hasta 4 de julio.
- Sprint 4. 40 horas. 4 de julio hasta 11 de julio.
- Sprint 5. 40 horas. 11 de julio hasta 18 de julio.
- Sprint 6. 40 horas. 18 de julio hasta 25 de julio.
- Sprint 7. 64 horas. 25 de julio hasta 3 de agosto.

En la figura 3.2 se muestra como se han repartido las tareas en los Sprints al final del proyecto.

Nº	Tareas	Tiempo (h)	Dependencias
1	**Desarrollo de la propuesta tecnica**	**36**	
1.1	**Inicio**	**8**	
1.1.1	Definir el contenido del proyecto	3	
1.1.2	Definir las tecnologias utilizadas	4	1.1.1
1.1.3	Definir las posibles mejoras para el proyecto	1	1.1.2
1.2	**Planificar el proyecto**	**28**	
1.2.1	Definir las tareas y estimar fechas	4	1.1
1.2.2	Crear diagrama de Gantt	8	1.1
1.2.3	Documentar la propuesta del proyecto	16	1.2.1, 1.2.2
1.2.4	Subir la propuesta tecnica	0	1.2.3
2	**Desarrollo tecnico del proyecto**	**340**	
2.1	**Desarrollo**	**340**	
2.1.1	Sprint 1	60	2.1
2.1.1	TU01, TU02, TU03, TU04, TU05, TU08, TU09 y TU10		
2.1.2	Sprint 2	40	2.1.1
2.1.2	TU06 y TU14		
2.1.3	Sprint 3	40	2.1.2
2.1.3	TU12		
2.1.4	Sprint 4	40	2.1.3
2.1.4	TU12		
2.1.5	Sprint 5	40	2.1.4
2.1.5	TU07		
2.1.6	Sprint 6	40	2.1.5
2.1.6	TU06 y TU07		
2.1.7	Sprint 7	80	2.1.6
2.1.7.1	TU07		
2.1.7.2	Puesta en marcha	20	
2.1.7.2.1	Implementación	16	2.1
2.1.7.2.2	Formación	4	2.1
2.1.7.2.3	Entrega final	0	2.2.1, 2.2.2
3	**Documentacion y presentación del TFG**	**145**	
3.1	Redaccion de informes quincenales	4	
3.2	Redaccion de la memoria tecnica	120	
3.3	Entrega de la memoria tecnica	0	
3.4	Preparacion de la presentacion oral	20	
3.5	Presentacion oral	1	

Figura 3.2: Estructura de descomposición del trabajo con las tareas realizadas.

Capítulo 4

Análisis y diseño del sistema

Este capítulo, en la sección 4.1, analiza los requisitos necesarios para realizar este proyecto. En la sección 4.2 se describe el diseño de los datos y las modificaciones que se han realizado en las bases de datos que aportan las dos plataformas. En la sección 4.3 se encuentra el diseño de la arquitectura donde se explica la arquitectura de los archivos de ambas plataformas. En la sección 4.4 se muestra una lista de las modificaciones que son necesarias. Y por último, en la sección 4.5 se analiza el diseño de la interfaz.

4.1. Análisis del sistema

En esta sección se especifican los requisitos necesarios para este proyecto. Los requisitos puede ser de dos tipos, funcionales y no funcionales. Los requisitos funcionales representan lo que se espera de la funcionalidad que el usuario u otro rol perciba en la aplicación final. Por otro lado, los requisitos no funcionales son características que tiene que tener el sistema y el usuario no percibe.

4.1.1. Requisitos funcionales

El cuadro 4.1 especifica los requisitos funcionales que se han obtenido con la técnica de las historias de usuario.

El cuadro 4.2 se desglosan las historias de usuario para mostrar las tareas que hay que realizar para conseguir cada historia de usuario. Esto ayuda a comprender en mayor profundidad los requisitos ya que se especifica mejor cada parte. Se ha utilizado puntos de historia para otorgar un valor aproximado a la dificultad de cada tarea. Cada punto de historia de usuario tiene un valor de 4 horas de trabajo.

Código	**Historia de usuario**
HU01	Como ingeniero informático quiero familiarizarme con el entorno de trabajo
HU02	Como ingeniero informático quiero aprender a usar las tecnologías y lenguajes de programación para desarrollar el proyecto.
HU03	Como administrador quiero administrar toda la información del portal web.
HU04	Como administrador quiero que las cuentas de cliente del e-commerce y de la plataforma formativa sean las mismas
HU05	Como profesor quiero crear y administrar mis cursos.
HU06	Como estudiante quiero comprar los cursos y acceder a ellos
HU07	Como administrador quiero que la forma de pago sea automática y al confirmar la compra de algún curso se matricule automáticamente al alumno
HU08	Como administrador quiero que el portal web tenga un buen posicionamiento en los motores de búsqueda
HU09	Como administrador quiero anunciarme a través de las redes sociales

Cuadro 4.1: Historias de usuario.

Id. historia de usuario	**Id. Tarea**	**Tarea**	**Puntos**
HU01	TU01	Familiarizarme con el entorno de trabajo	1
HU02	TU02	Aprender los lenguajes de programación y plataformas	6
HU03	TU03	Crear cuenta de administrador	0
	TU04	Comprobar los permisos de la cuenta de Administrador	0
HU04	TU05	Crear cuenta de usuario en la plataforma formativa y e-commerce	1
	TU06	Sincronizar cuentas de usuario en la base de datos	6
	TU07	Sincronizar la identificación de las cuentas	20
HU05	TU08	Crear cuenta de Profesor y otorgarle permisos	1
HU06	TU09	Comprar cursos	0
	TU10	Acceder a los cursos	0
HU07	TU11	Proporcionar y configurar una pasarela de pago	8
	TU12	Matricular al alumno en el curso adquirido	10
HU08	TU13	Proporcionar SEO	4
HU09	TU14	Facilitar anunciarse a través de las redes sociales	4

Cuadro 4.2: Estimación en puntos de historia de las historias de usuario del proyecto.

4.1.2. Requisitos no funcionales

Estos requisitos no añaden una funcionalidad al sistema y el usuario no es necesario que conozca dicha funcionalidad. Los requisitos no funcionales se presentan a continuación:

- La parte de la venta de cursos online tiene que utilizar la plataforma Prestashop.
- La parte de creación de cursos tiene que usar la plataforma Moodle.
- El uso de la aplicación debe ser intuitivo, para usuarios que no estén habituados a esta tecnología.
- La aplicación debe ser robusta y fiable.

4.2. Diseño de los datos

Tras realizar una instalación, la base de datos que instala Prestashop contiene una gran cantidad de tablas, aproximadamente 160 [1]. Se destaca las tablas que tienen relación con el proyecto, por tener relación o bien con el producto o bien con el usuario. En la figura 4.1 se muestra los campos que contiene cada tabla:

- **ps_customer**. Guarda la información que envía el usuario al crear una cuenta.
- **ps_product**. Guarda la información básica de los productos.
- **ps_cart_product**. Guarda los productos que se almacenan en cada carrito de la compra.
- **ps_orders**. Guarda los pedidos que realizan los usuarios.

Moodle por defecto instala una base de datos con más de doscientas tablas [2], que son necesarias para el *core* y los *plugins* principales. Las tablas de la base de datos que se han utilizado en este proyecto son:

- **m_users**. Contiene los detalles básicos de cada usuario.
- **m_course**. Contiene la información de los cursos.
- **m_context**. Esta tabla define el 'contexto' [3]. El 'contexto' se refiere a qué alcance tienen los permisos en Moodle, por ejemplo en todo el sistema, un curso o una actividad en concreto.

[1] `http://preprod-img.prestashopinc.netdna-cdn.com/blog/articles-blog/2012-04-04/prestashop-datamodel.png`

[2] `http://www.examulator.com/er/2.2/moodle22_erd.png`

[3] `https://docs.moodle.org/29/en/Context`

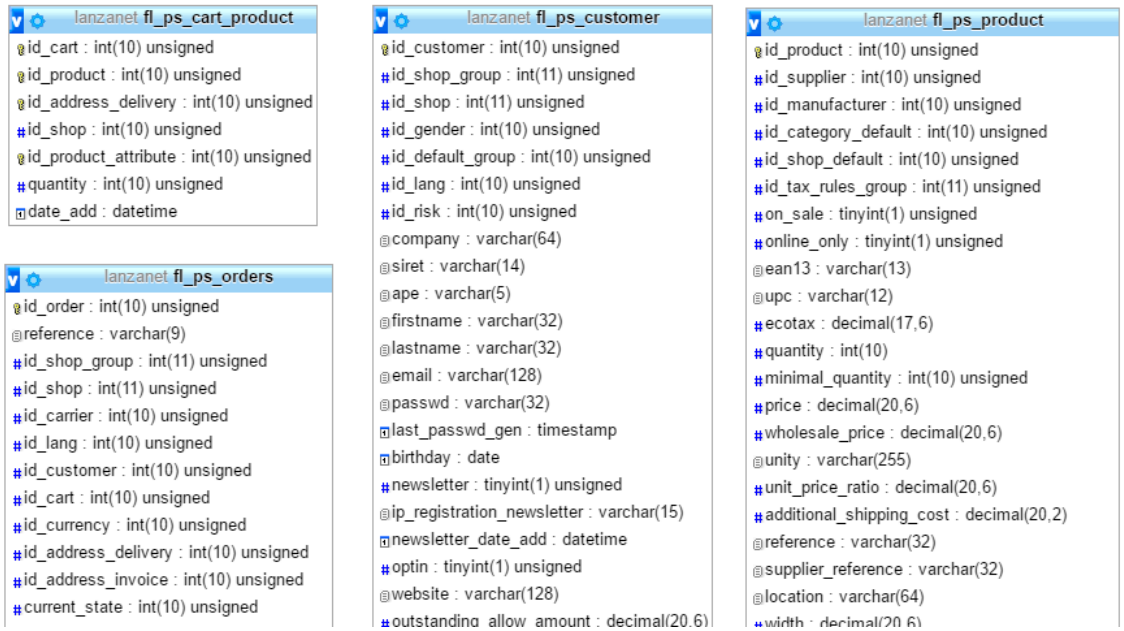


Figura 4.1: Tablas utilizadas de la base de datos de Prestashop.

- **m_enrol**. Guarda la información sobre los métodos de matriculación que se pueden realizar por cada curso.
- **m_user_enrolments**. Guarda los usuarios matriculados en los cursos.
- **m_role_assignments**. Indica a qué usuario se le asignan roles según el 'contexto'.

El nombre de los campos de las tablas se muestra en la figura 4.2.

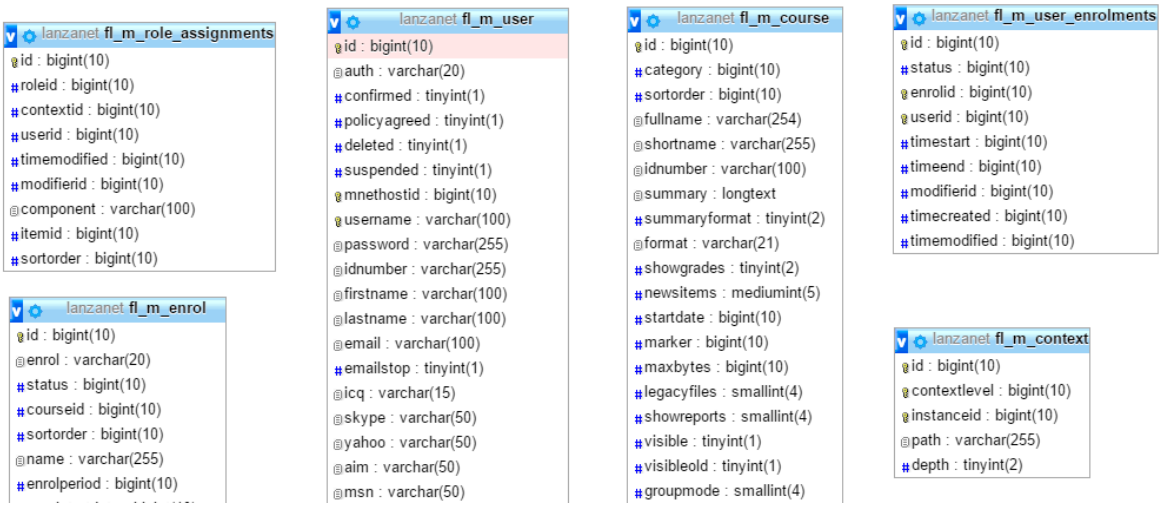


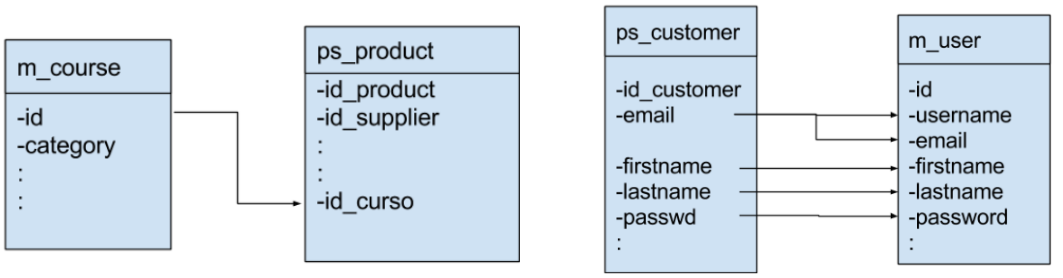
Figura 4.2: Tablas utilizadas de la base de datos de Moodle.

Para cubrir la necesidad que desde los productos de Prestashop se pueda acceder a la información de los cursos de Moodle, se ha añadido un nuevo campo en la tabla **ps_product** al

que se ha nombrado como *id_curso*. Dentro de *id_curso* se guardará el valor del campo *id* de la tabla **m_course**. Gracias a esto se conoce a qué curso de Moodle hay que matricular a los alumnos que hayan comprado un curso a través de Prestashop.

Para un correcto funcionamiento de la plataforma es necesario que se comparta la misma cuenta de usuario en Moodle y en Prestashop y para ello se ha configurado Moodle para que recoja de la tabla de Prestashop *ps_customer* los campos necesarios para poder utilizar una cuenta de usuario en Moodle. Para ello hay que indicar algunos campos y el nombre de la tabla donde están los datos de los usuarios en la configuración de identificación usando una tabla externa en Moodle.

En la figura 4.3 se pueden observar dos esquemas con las conexiones entre las tablas y los campos de ambas bases de datos.



(a) Se guarda el id de un curso en un producto

(b) Se iguala algunos valores de ps_customer en m_user

Figura 4.3: Conexión de las tablas de las bases de datos de Prestashop y Moodle.

En la figura 4.4 se muestra la estructura simplificada de las tablas de ambas bases de datos y las conexiones que tienen entre ambas. Cabe destacar que las tablas mostradas son las que se han explicado anteriormente y que se han omitido el resto de las tablas de ambas base de datos y sus relaciones por simplificar el esquema.

4.3. Diseño de la arquitectura del sistema

Prestashop sigue el patrón Modelo-Vista-Controlador (MVC) como arquitectura de software. Tras realizar una instalación se crean diversas carpetas y entre estas se observan las que forman parte de la arquitectura MVC:

- La carpeta **/classes** corresponde a la parte del Modelo. Dentro se encuentra un archivo por cada tabla en la base de datos. Cada archivo contiene los atributos de cada tabla de la base de datos y funciones que utilizan dichos atributos.
- La carpeta **/controllers** corresponde a la parte del Controlador. Y dentro se encuentran dos carpetas, una para el *frontoffice* y otra para el *backoffice*. En ambas carpetas hay archivos que contienen métodos para procesar las peticiones que se reciben desde la Vista.

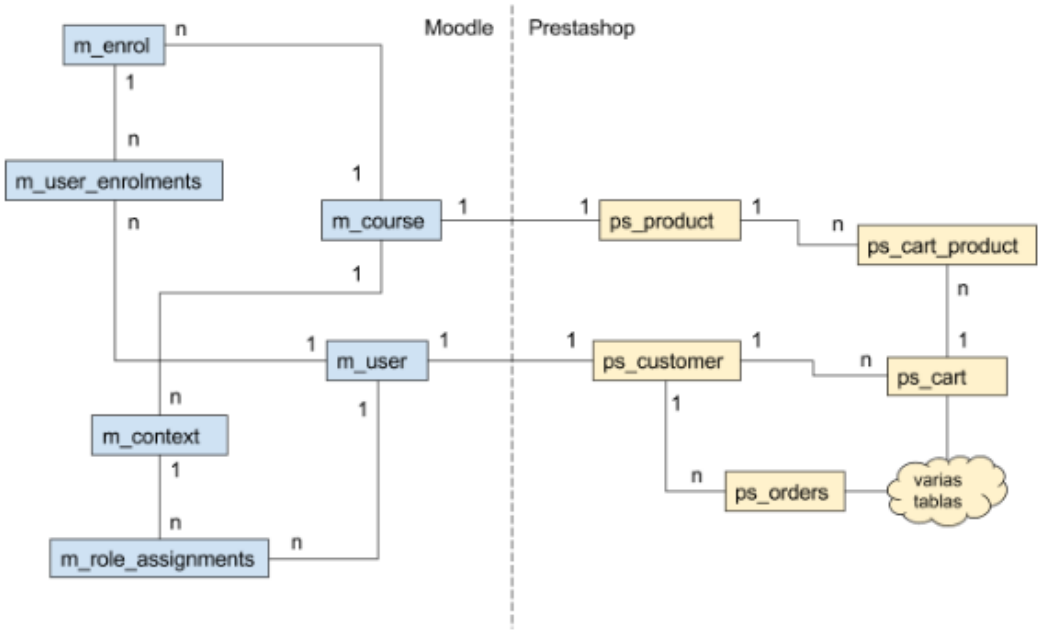


Figura 4.4: Estructura simplificada de las bases de datos de ambas plataformas.

También es común encontrar dentro de los métodos una línea que activa los métodos asociados a un *Hook* después de un evento predeterminado.

- La carpeta **/themes** corresponde a la parte de Vista. Contiene los archivos correspondientes a las plantillas que se utilizan para generar las vistas. Son archivos que contienen tanto código html como código php. En Prestashop se utiliza un framework llamado Smarty para facilitar obtención de los valores de las variables que se envían desde los controladores.
- Las carpetas **/js** y **/css** también corresponden a la parte de Vista. Aquí se almacenan archivos .css y .js que configuran el aspecto visual.

Los *Hooks* [4] que se han comentado en los controladores son un punto donde se asocia un código a ciertos eventos. Existen dos tipos de *Hooks*, los de tipo acción (Action) o para mostrar algo (Display). Los *Hooks* de tipo mostrar son los encargados de la estructura de las páginas, su principal función es devolver un código html para que se muestre en una posición determinada. Los tipo acción actúan después de que ocurra un evento, por ejemplo se añade un método o módulo en el *Hook* que se ejecuta después de la creación de una cuenta para que envíe un correo dando la bienvenida.

Un *Hook* se trata de una implementación del patrón *Observer*. Los eventos a los que está asociado un *Hook* se definen en los controladores y clases de entidad, haciendo una llamada a todos los módulos asociados a ese *Hook*. Se puede enviar además unos parámetros, que por defecto son el carrito de compra y las cookies, aunque se pueden modificar para que reciban más parámetros.

Otra carpeta importante dentro de Prestashop es la carpeta **modules** donde se almacenan los distintos módulos que están instalados en nuestra tienda Prestashop. Estos módulos añaden alguna funcionalidad nueva o bien modifican el comportamiento de algún apartado. Los módulos

pueden modificar cada parte de nuestra tienda ya que pueden añadir nuevos métodos en los *Hooks*.

La arquitectura que utiliza Moodle es conocida como *Moodle core* [4] que consiste en un núcleo al que se le añaden varios *plugins*. No está orientada a objetos ya que está programada en *legacy PHP*. Por defecto el *core* tiene mecanismos necesarios para implementar la plataforma educativa y sus componentes son los cursos, los usuarios y las matriculaciones. Los *plugins* añaden funcionalidades a estos componentes y los más importantes son los que se encuentras en las siguientes carpetas:

- **/auth**. Contiene *plugins* para realizar diferentes tipos de identificación de usuarios.
- **/mod**. Esta carpeta contiene las distintas actividades que se pueden realizar dentro de los cursos. La comunidad de Moodle ha desarrollado más *plugins* en este ámbito.
- **/enrol**. Contiene los *plugins* de las matriculaciones.
- **/theme**. En esta carpeta se almacena los estilos de las páginas.

Una de las características de la arquitectura de Moodle es que los *plugins* son modulares, permitiendo que se puedan añadir modificaciones al código base con facilidad.

4.4. Modificaciones necesarias

Para conseguir los objetivos propuestos, además de las modificaciones indicadas en la sección anterior, se necesita adecuar distintos elementos de la funcionalidad de Prestashop y Moodle:

- **Eliminar la dirección de entrega**. Prestashop necesita de una dirección para completar un pedido pero al tratarse de productos virtuales este campo es innecesario, de modo que se ha de eliminar de los formularios y facturas.
- **Modificar el modelo de los productos de Prestashop**, para incluir el nuevo campo que identifica el curso de Moodle *id_curso* en la tabla de productos de Prestashop *ps_product*.
- **Usar las mismas cuentas de usuario en ambas plataformas**. Se ha de configurar Moodle para usar la tabla *ps_customer* de la base de datos de Prestashop para la identificación de usuarios.
- **Matriculación de cursos en Moodle desde Prestashop**. Después de que un *estudiante* compre un curso en Prestashop, debe aparecer matriculado en Moodle para acceder a su contenido. En la figura 4.5 se ve el esquema para obtener el identificador de curso y usuario de Moodle para después insertarlos en las tablas necesarias para que Moodle reconozca las matriculaciones, como se ve en la figura 4.6.

[4] `https://docs.moodle.org/dev/Moodle_architecture#An_overview_of_Moodle_core`

- **Modificación del botón comprar**. Cuando el usuario ya ha adquirido un curso, en lugar del botón comprar se ha de mostrar un mensaje indicando que ya está matriculado en este curso.
- **Conexión con las redes sociales**, principalmente Facebook y Twitter para facilitar el seguimiento de la plataforma en dichas redes sociales.

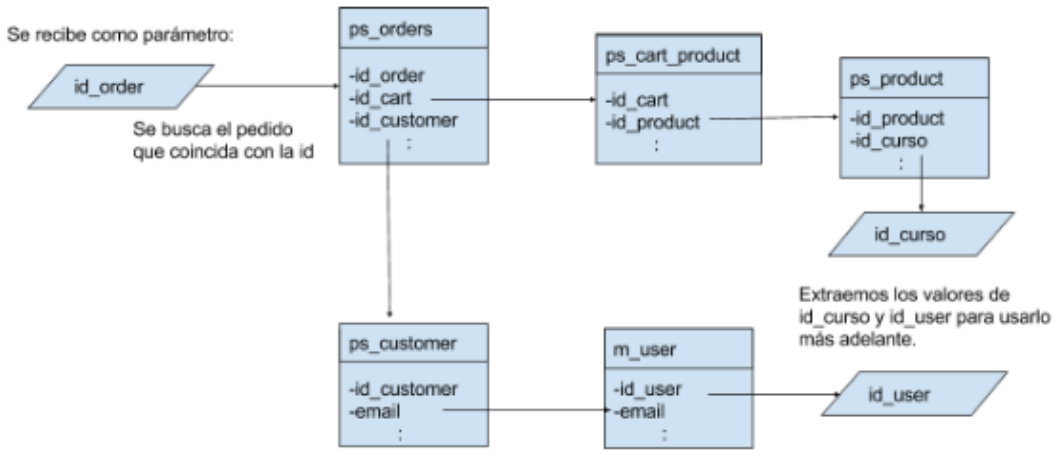


Figura 4.5: Matriculación. Obtención del identificador de curso y de usuario a partir del identificador del pedido.

4.5. Diseño de la interfaz

Esta sección explica las interfaces de usuario que aporta por defecto Prestashop y Moodle y las modificaciones que se han realizado para conectar ambas plataformas.

Prestashop y Moodle traen una interfaz funcional y adaptable a la pantalla del usuario. En las figuras 4.7 y 4.12 se ven las páginas iniciales tras realizar la instalación.

Se quiere hacer que el frontoffice de Prestashop sea la pantalla principal que vean los alumnos cuando entren a la plataforma. Aquí se mostrarán los cursos que están a la venta a través de un carrusel de imágenes que redireccionan a los detalles de los cursos si se pulsan en las imágenes, como se ve en la figura 4.8. En la figura 4.9 se ven los detalles de los cursos y se muestra una pequeña descripción, otra descripción más completa y el botón de comprar, cuyo comportamiento se ha modificado. El botón de comprar únicamente estará disponible si no se ha comprado el curso y en caso de que se haya adquirido aparecerá un mensaje indicado que ya tiene dicho curso.

Una de las modificaciones que se han realizado a la interfaz de Prestashop es el hecho de que no aparezcan las direcciones porque los productos que se venderán son virtuales y no requieren

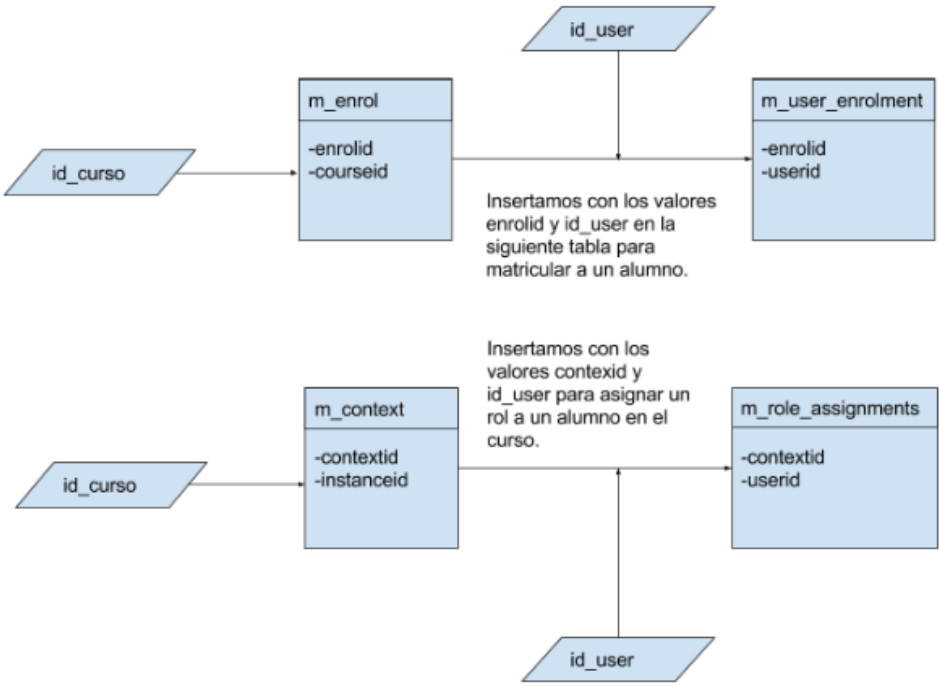


Figura 4.6: Matriculación de un estudiante en un curso. Inserciones necesarias en la base de datos de Moodle.

de una direccion física para entregar el producto. Por ello se han eliminado las direcciones de los formularios de creación de cuentas, en las confirmaciones de los pedidos y en las facturas que se generan al confirmar los pedidos. En la figura 4.10 se observa que no aparecen los campos en los formularios donde se indica las direcciones de entrega.

En la figura 4.11 se ve la interfaz del backoffice de Prestashop, la cual no ha recibido cambios significativos. Únicamente se ha modificado el formulario de los productos para añadir un campo en el que se puede editar la identificación de los cursos de Moodle que tienen almacenado.

Figura 4.7: Interfaz básica del frontoffice de Prestashop.

Figura 4.8: Interfaz modificada del frontoffice de Prestashop.

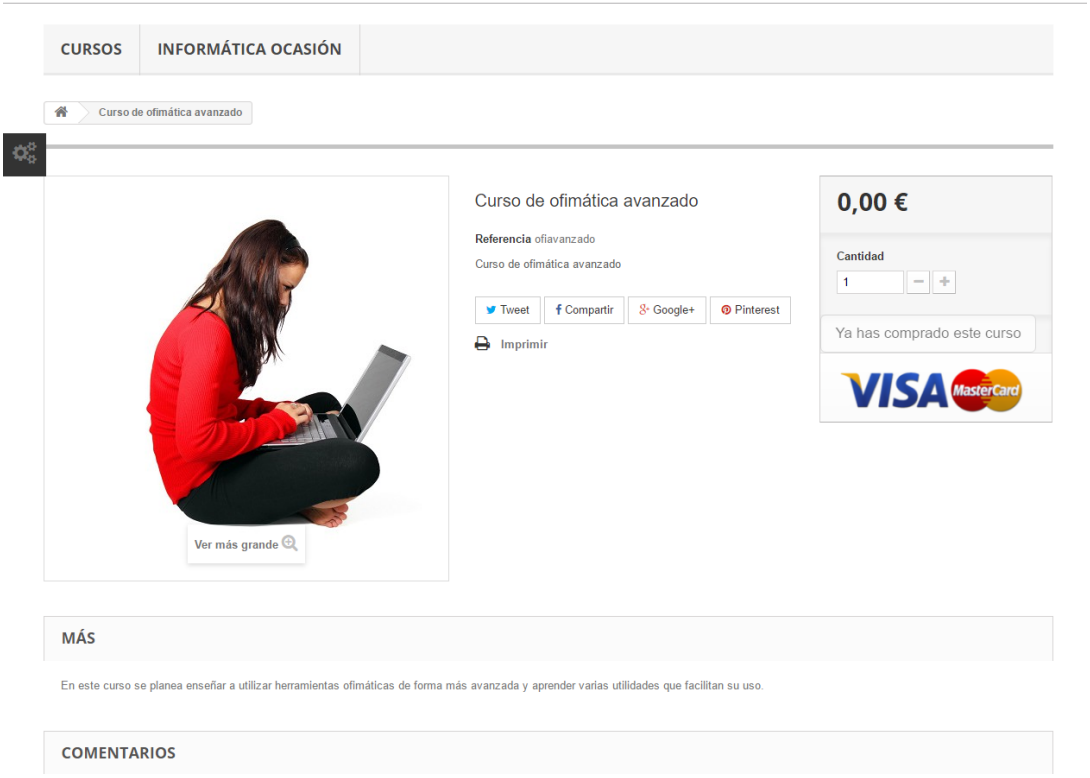
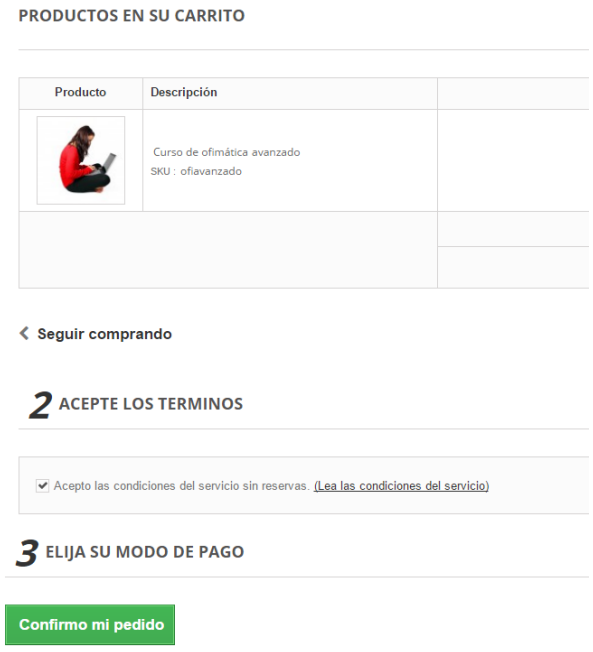


Figura 4.9: Detalles de un curso en Prestashop.

(a) Formulario de creación de cuentas



(b) Confirmación de pedido

Figura 4.10: Modificaciones a los formularios para no mostrar la dirección de entrega.

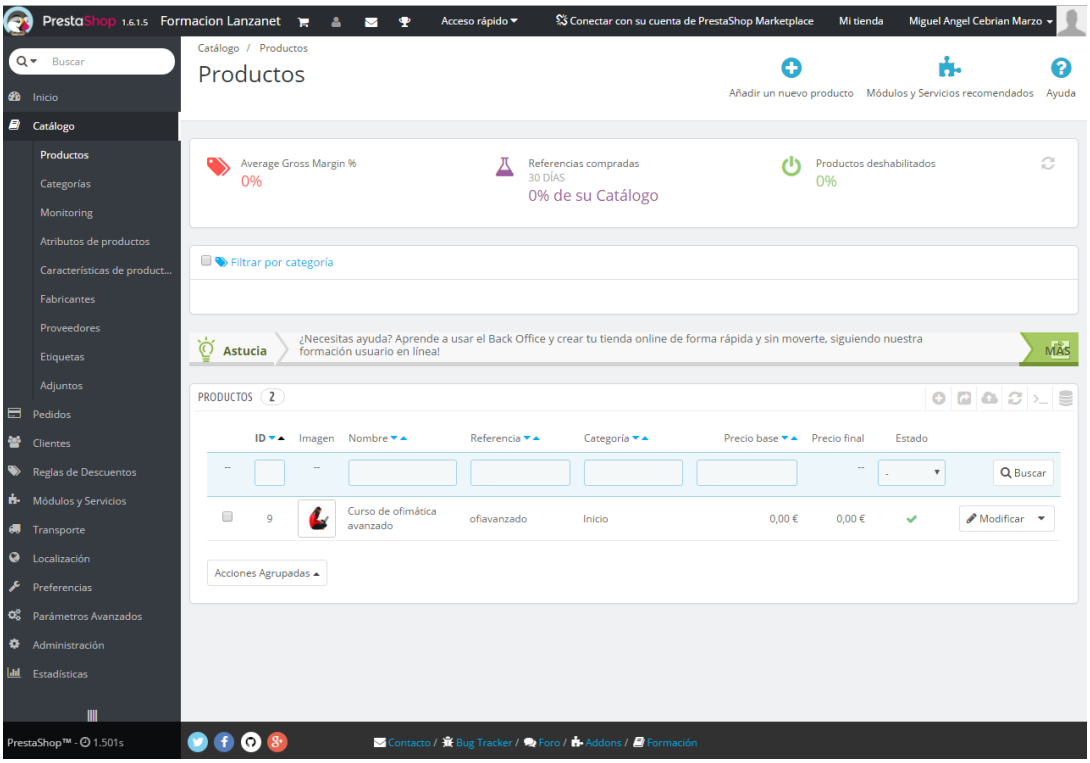


Figura 4.11: Backoffice de Prestashop.

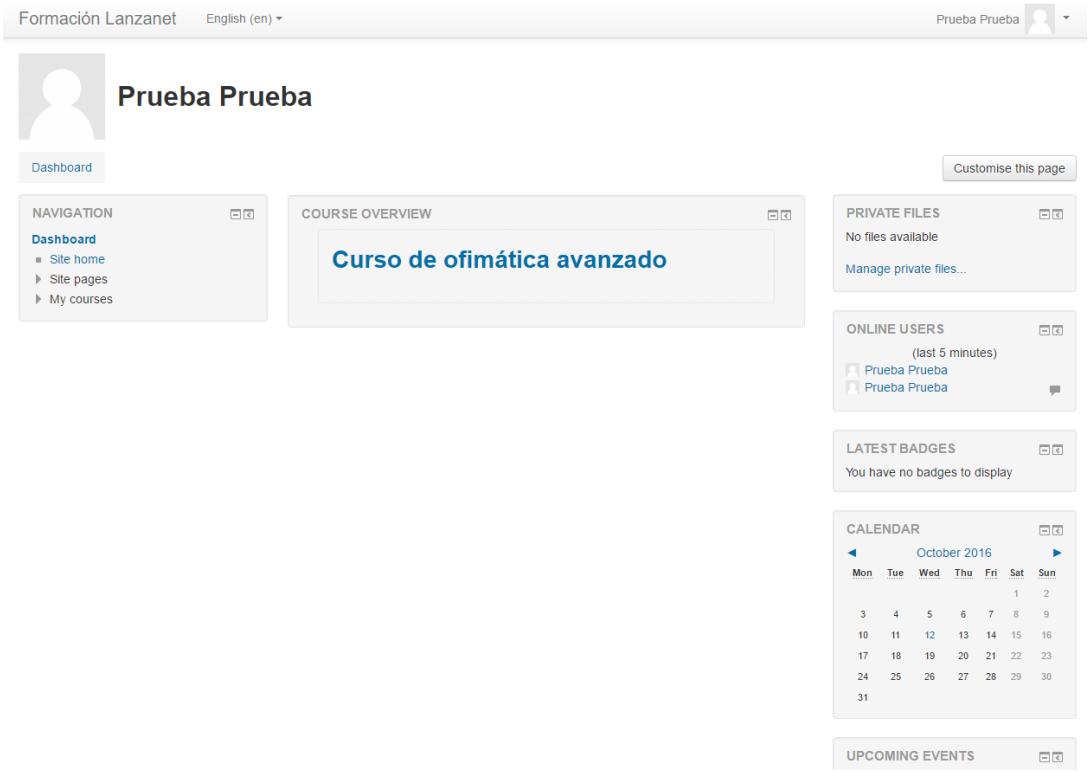


Figura 4.12: Interfaz Moodle.

Capítulo 5

Implementación

En este capítulo se explican los procesos que se llevaron a cabo durante el desarrollo de este proyecto. En la sección 5.1 se explica en detalle cada modificación que se ha realizado en ambas plataformas. En la sección 5.2 se explica que se ha modificado en los archivos a la hora de subir el proyecto a un servidor externo y en la sección 5.3 se muestra cómo se han realizado las tareas en cada Sprint.

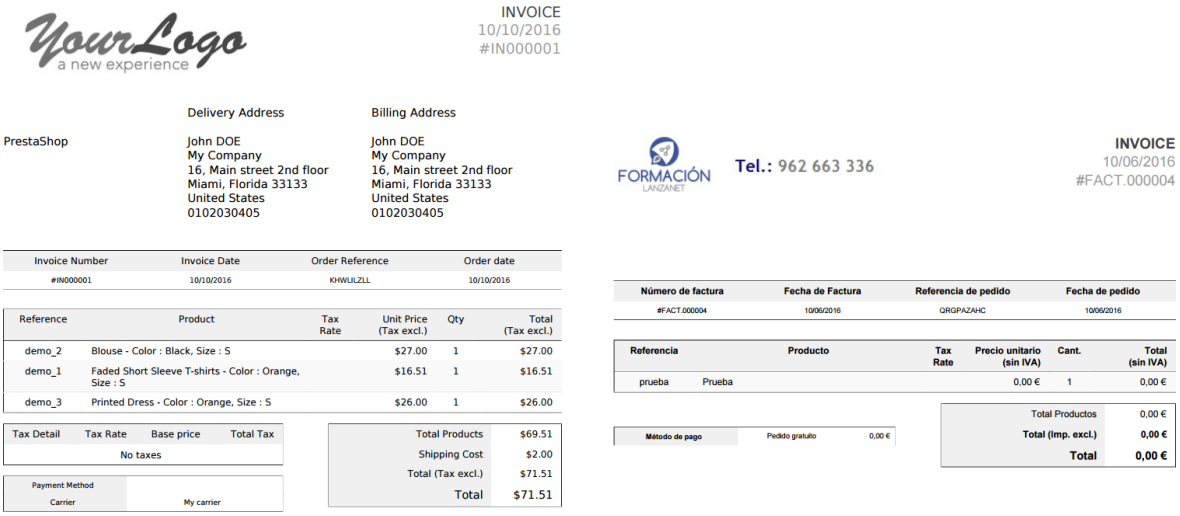
5.1. Detalles de implementación

En cada sección se explica en detalle los elementos más importantes que se han desarrollado para este proyecto, con una descripción indicando los pasos que se han seguido para su desarrollo y una imagen con el resultado final, si la poseen.

5.1.1. Eliminación de la dirección de entrega

Una de las primeras modificaciones que se ha realizado es que al realizar un pedido no se necesitase de una dirección de entrega [5]. Dado que se está trabajando con productos virtuales este campo es innecesario. Hay que eliminar los campos de dirección tanto en el formulario cuando se realiza un pedido de productos, en el formulario de creación de cuentas de usuario y en la factura que se crea después de que un pedido confirme su pago, como se puede apreciar en la figura 5.1. Aunque en las últimas versiones de Prestashop se ha incluido marcar un producto como virtual y así no ser necesario indicar una dirección al realizar un pedido, sigue apareciendo en el formulario de creación de cuentas y en la factura.

Aunque se haya suprimido los campos de la dirección de entrega, Prestashop sigue necesitando de una dirección para poder completar un pedido. Para solucionarlo, se ha optado por crear un campo oculto en el formulario que envíe una dirección falsa. Esta solución [5] no es óptima ya que se crean datos repetidos por cada usuario creado.



(a) Factura básica de Prestashop

(b) Factura modificada sin la dirección

Figura 5.1: Diferencia entre facturas.

5.1.2. Modificación del modelo de los productos de Prestashop

Al realizarse la modificación en la tabla **ps_product** para añadir el campo con el nombre *id_curso* que represente al identificador del curso de Moodle es necesario que se modifique la clase que actúa como el modelo de la tabla para que Prestashop reconozca este nuevo campo. Gracias a esto se puede acceder a la nueva información desde la administración de un producto y que pueda ser guardada en la tabla de los productos de Prestashop. Esta modificación permite acceder a la identificación de un curso de Moodle y poder relacionar un producto de Prestashop con un curso de Moodle. Sin esta modificación no podrá matricular a un usuario según el curso que adquierá en Prestashop.

Dado que se ha modificado la tabla **ps_product** hay que añadir nuevas líneas de código en el archivo */classes/Product.php* para el nuevo campo y sus características. Las clases Modelo en Prestashop heredan de una clase abstracta llamada *ObjectModel* y contiene métodos ya hechos que facilitan añadir nuevos campos. Únicamente se crea una variable nueva con el mismo nombre que el campo y se añade en la lista donde se indican todos los campos y sus tipos, que en este caso se asigna como un entero. La información que se guarda en la lista se puede observar en la figura 5.2. Gracias a esto se puede modificar y acceder a todos los valores de la tabla de la base datos utilizando el backoffice que proporciona Prestashop.

```
//ADDED
'id_curso' =>                array('type' => self::TYPE_INT, 'lang' => true, 'validate' => 'isUnsignedId', 'size' => 10),
'associations' => array(
```

Figura 5.2: Línea de código donde se añade al modelo de Customers el identificador del curso de Moodle.

Ahora se necesita poder modificar la información que se guarda de un producto en Prestashop

para indicar a qué curso de Moodle hace referencia. Se necesita añadir en el formulario de la administración de los Productos una nueva pestaña con un campo donde se puedan seleccionar los cursos de Moodle que aún no han sido asignados.

5.1.3. Identificación de usuarios

Una de las modificaciones en la configuración de Moodle es la identificación de usuarios [6]. Conectado como *Administrador*, se puede configurar la identificación dentro de Administración del sitio, en la parte de Extensiones e Identificación. Una de las distintas opciones que permite a Moodle identificar a los usuarios es utilizar una base de datos externa para obtener los datos de los usuarios. Gracias a esto se puede mantener el mismo usuario tanto en Moodle y en Prestashop, evitando crear dos cuentas de usuario por cada usuario real. A la hora de comprobar los datos para identificar a los usuarios se utiliza la tabla de la base de datos que se le indique. También hay que indicar algunos de los campos de la tabla externa que corresponde a los campos de la tabla de usuarios de Moodle, como por ejemplo el nombre de usuario o la contraseña.

En este caso, la tabla de usuarios de la base de datos de Prestashop es *ps_customer*, el campo que corresponde al nombre de usuario es *email* y el campo de la contraseña es *passwd*. En la figura 5.3 se ve cómo configurar la identificación con la base de datos de Prestashop.

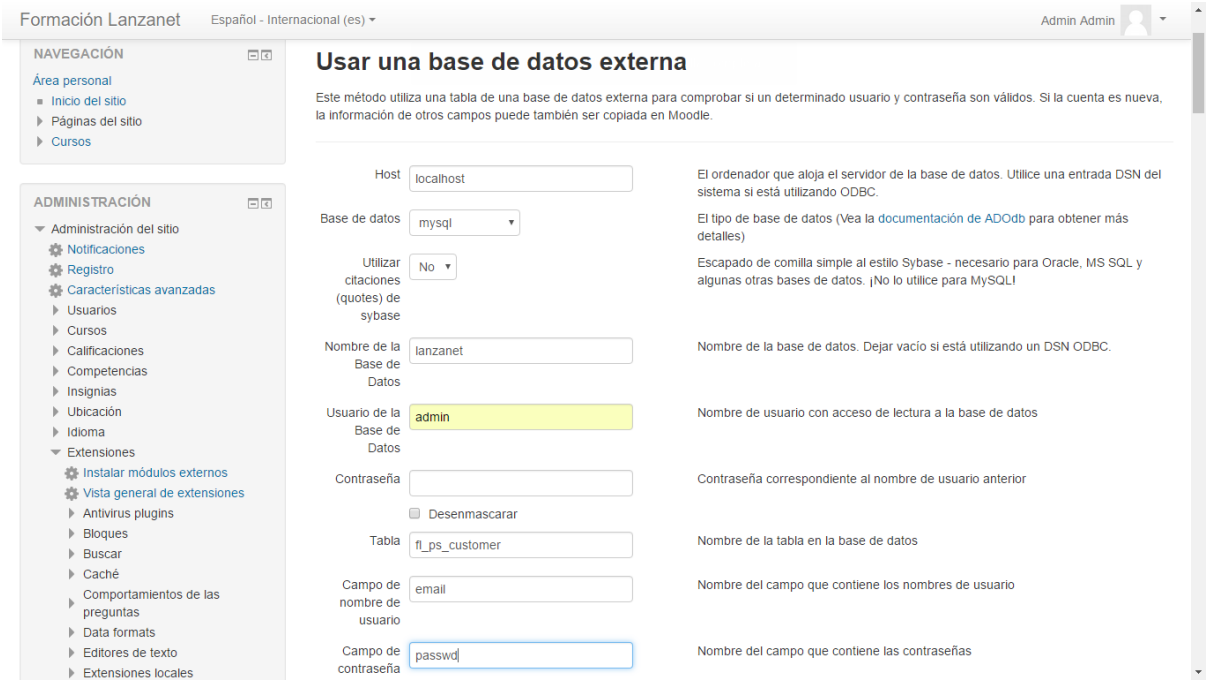


Figura 5.3: Configuración de la identificación con la base de datos de Prestashop.

Se ha estudiado cómo realizar la sincronización de las cuentas, esto es, que cuando un usuario en Prestashop se identifique también lo haga en Moodle ya que comparten el mismo usuario y contraseña. Así se evita que un usuario tenga que introducir dos veces su usuario y contraseña para poder acceder a ambas partes de la plataforma. Durante el tiempo que se ha realizado este proyecto se ha implementado varias soluciones para conseguir la sincronización con los métodos

tanto Single Sign-On [1] como OpenID [2]. Pero no se ha conseguido que funcionen correctamente, por lo que se ha optado por el método anterior aunque resulte más tedioso para el usuario. En la sección 5.2, Sprint 5, se explica con detalle las prueblas realizadas.

5.1.4. Codificación de la contraseña

En la base de datos de Prestashop se ha modificado el valor guardado en el campo *passwd* dentro de **ps_customer** para añadir un Salt [7] a las contraseñas. Se ha modificado el método que guardaba la contraseña en la base datos para que se codifique la contraseña del usuario junto a la variable *secure_key* con el algoritmo *md5* [3] para evitar ataques comunes, como por ejemplo de fuerza bruta o de diccionario. El valor de la variable *secure_key* es un valor aleatorio que se ha codificado con el algoritmo *md5*. También se ha modificado parte del método *getByEmail* que se utiliza para identifica a los usuarios a través del correo electrónico y una contraseña y se aplica la misma codificación para poder comparar las contraseñas.

```
//Añadir el Salt para codificar la contraseña
$this->passwd = md5($this->passwd.$this->secure_key);
```

Figura 5.4: Codificación de contraseñas.

Como se ha modificado la codificación de contraseñas en Prestashop, se realiza el mismo proceso en Moodle a la hora de comprobar las contraseñas para identificar el usuario. En el fichero **moodle/auth/db/auth.php**, dentro del método que realiza la comprobación, se recoge el valor del campo *secure_key* de la tabla *ps_customer* del usuario que coincida el *email*. Se codifica este valor junto a la contraseña que envía el usuario y se comprueba que coincide con el valor que está guardado en *passwd* de la tabla *ps_customer* para identificar al usuario.

5.1.5. Conexión con las redes sociales

Una de las funcionalidades necesarias era integrar las redes sociales, principalmente en Facebook y Twitter, en nuestra tienda online Prestashop para poder publicitar nuestra plataforma utilizando estas redes sociales. Gracias a que ya existen módulos de Prestashop que realizan esta tarea se evita crear los módulos desde cero, ahorrando tiempo y se asegura que funcionen correctamente.

Para conectarse con Facebook se ha utilizado un módulo gratuito llamado **Prestashop Facebook Like Box**. Gracias a este módulo se añade fácilmente en nuestra página principal de Prestashop un *plugin* que muestre un enlace a nuestra página en Facebook, el número de personas que la sigue y un botón para poder seguir la página de Facebook. Este plugin necesita que se indique en la configuración la página de Facebook.

Para conectar Twitter se ha utilizado otro módulo gratuito llamado **Prestashop Twitter widget**. Con este módulo se muestran los últimos tweets que se han enviado a través de nuestra

[1] https://es.wikipedia.org/wiki/Single_Sign-On
[2] https://es.wikipedia.org/wiki/OpenID
[3] https://es.wikipedia.org/wiki/MD5

cuenta de Twitter y además añade un botón que facilita seguir nuestra cuenta de Twitter. Para configurar este módulo requiere que se indique cual es nuestra cuenta de Twitter y una identificación llamada *Twitter Widget Id* [8] que se obtiene creando un nuevo widget con nuestra cuenta de Twitter.

En la figura 5.5 se puede observar los módulos una vez instalados y configurados.

Figura 5.5: Plugins de las redes sociales.

5.1.6. Módulo ConectaMoodle

Matriculación de cursos en Moodle desde Prestashop

Para este proyecto se ha creado un módulo de Prestashop que realiza alguna de las tareas necesarias para interconectar ambas plataformas utilizando la tecnología de los *Hooks* [4]. Este módulo pretende enlazar las dos plataformas, Prestashop y Moodle, añadiendo métodos en los *Hooks* para que cuando se activen determinados eventos realicen unas tareas que sincronicen ambas plataformas. En la figura 5.6 se observa la instalación del módulo en el instalador de módulos de Prestashop.

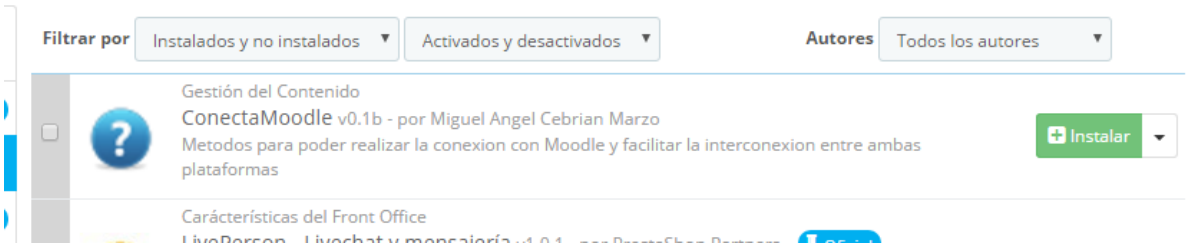


Figura 5.6: ConectaMoodle en la ventana de instalación de módulos de Prestashop.

Los *Hooks* que se han utilizado para desarrollar este módulo son los siguientes:

- El Hook **ActionPaymentConfirmation**. Este *Hook* se activa después de que se confirme el pago de una compra. Se utilizará para realizar la matriculación del alumno en el mismo curso de Moodle que se ha comprado.
- El Hook **DisplayAdminProductsExtra**. Este *Hook* muestra una pestaña adicional en la configuración de un producto y dentro de la pestaña añadirá una ventana con nuevo un campo donde se asignará un curso de Moodle a un producto de Prestashop.
- El Hook **ActionProductUpdate**. Este *Hook* se activa cuando se va a guardar la información de un producto. Se utilizará para guardar los nuevos datos que se obtienen del nuevo del anterior *Hook*.

Una de las taras que realiza este módulo es que en el momento que se confirma el pago de una compra, se matricule al usuario en el curso que se ha comprado. Para realizar esta tarea se añade al *Hook* ActionPaymentConfirmation un método, que inserta nuevas líneas en distintas tablas de la base de datos de Moodle, que son necesarias para que Moodle detecte un usuario matriculado en un curso.

Al método que se añade dentro del *Hook* recibe como parámetro los valores del pedido, de estos valores se obtiene el *id_order* que representa la identificación del pedido que se acaba de confirmar. Con este valor se obtiene todos los productos (cursos) que se han comprado y en cada producto se guarda la identificación de cada curso en Moodle. Se obtiene una lista con las identificaciones de los cursos que hay que matricular al usuario. También se obtiene la identificación de usuario en Moodle a través de *id_order*. La figura 4.5 representa cómo se obtienen los valores de las distintas tablas de la base de datos.

Moodle posee varias formas de matricular a un alumno, por ejemplo añadir manualmente a un alumno desde la ventana de administración o bien a través de un enlace que se puede enviar a través de alguna via, por ejemplo correo electrónico, para que el interesado se matricule pulsando el enlace. Se realizaron varias pruebas para poder añadir a un alumno desde Prestashop pero no fue posible ya que se limitaba el uso de los métodos de Moodle, por lo que se ha emulado la matriculación añadiendo unas líneas en la base de datos de Moodle como si de una matriculación manual se tratase.

Para poder entender como funciona la matriculación en Moodle hay que conocer el funcionamiento de cada tabla y que se extrae de cada una para que Moodle reconozca a un estudiante matriculado en el curso que se desee [9]. En la figura 4.6 se muestra con un esquema este proceso. Ahora se utilizarán los valores que hemos recogido anteriormente para realizar dos búsquedas:

- Buscar el *id* de la tabla *m_enrol* donde el campo 'enrol' tenga como valor 'manual' y 'courseid' sea igual al *id* del curso. Con este valor se obtiene una variable que indica que método de matriculación se a utilizado para matricular a un alumno y es necesario en la tabla *m_user_enrolments* junto al *id* del alumno para que este alumno aparezca matriculado en un curso.

- Buscar el *id* de la tabla *m_context* donde el campo 'contextlevel' tenga como valor 50 e 'instanceid' sea *id* del curso. Es necesario que 'contextlevel' tenga como valor 50 ya que representa que el 'contexto' tenga un alcance de un curso. Los valores de 'contextlevel' son estáticos y van desde el valor 10 al 80[4], donde 10 representa a todo el sistema y 80 a un bloque dentro de un curso. Una vez se obtenga el *id* que contenga el 'contexto' del curso, se añade a la tabla *m_role_assignments* este valor junto al *id* del alumno para que este alumno obtenga el rol de *estudiante* en este curso.

Una vez se añaden las dos líneas en las tablas correspondientes se observa en el panel de administración que el alumno ha sido matriculado en el curso que se deseaba, mostrándose como una matriculación manual y con el rol de estudiante. Gracias a esto se realiza una matriculación desde fuera de Moodle siempre que se tenga acceso a la base de datos.

Modificación del formulario de productos

Este módulo también añade en el formulario de administración de los productos una pestaña adicional donde se modifica el valor de la identificación del curso de Moodle al que hace referencia ese producto. Gracias al *Hook* DisplayAdminProductsExtra carga una nueva pestaña en el formulario de administración de los productos y se puede añadir nuestro código html para que muestre un despegable con los cursos de Moodle que aun no han sido asignados a un producto de Prestashop. Para guardar los valores se utiliza el *Hook* ActionProductUpdate que actuá después de que un producto se ha modificado, y se actualiza el valor de la variable *id_curso* recogiendo su valor del formulario que se ha creado. En la figura 5.7 se observa el aspecto que tiene el formulario en la ventana de administración de los productos.

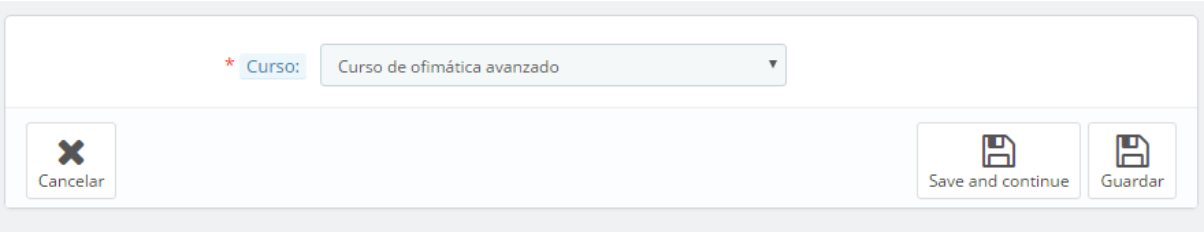
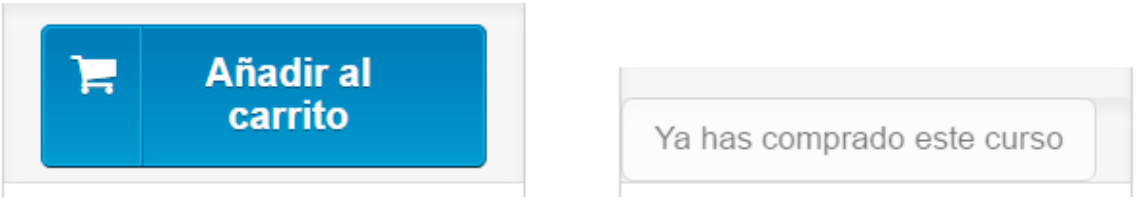


Figura 5.7: Formulario en la ventana de edición de Productos.

5.1.7. Modificación del botón de comprar

Como medida para evitar un problema que apareció al realizar un test, se ha modificado el comportamiento del botón 'comprar un curso' en el caso de que ya se haya adquirido el curso anteriormente. Se ha creado un método dentro del controlador de los Productos (ProductController) y comprueba si el usuario ha comprado este curso. En caso afirmativo mostrará un mensaje indicando que ya se ha matriculado en tal curso donde anteriormente estaba el botón de comprar el curso y bloquea que se pueda volver a comprar dicho curso. En la figura 5.8 se observa la diferencia cuando un usuario ha comprado un curso y cuando no.

[4] `https://docs.moodle.org/dev/Roles_and_modules#Context`



(a) Curso no comprado (b) Curso comprado

Figura 5.8: Botones de compra de un curso en Prestashop.

5.2. Puesta en marcha

Mientras se desarrollaba el proyecto se estaba realizando en un servidor local creado con XAMMP [5]. Una vez se ha subido el proyecto en el nuevo servidor, se modifica varios ficheros relacionados con la configuración de la base de datos en ambas plataformas.

Para configurar Prestashop se ha modificado los valores que se encuentran en **PrestaShop/config/setting.inc.php** por los valores de la nueva base de datos. También se modifica en la base de datos en las tablas **ps_shop_url** y **ps_configuration** para indicar la nueva dirección del servidor. Para evitar problemas se ha regenerado el fichero **.htaccess** [6] desde el backoffice de Prestashop.

Como Moodle no funcionaba correctamente, hubo que instalar desde cero Moodle en el nuevo servidor y volver a configurar y subir los ficheros que se habían modificado.

El contenido de la base de datos interna se exportó en un archivo *.sql* para poder importarla en la nueva base de datos.

5.3. Sprints

Esta sección se describirán los Sprints y el desarrollo del proyecto. Cada Sprint ha durado aproximadamente una semana, excepto el primero y el último. El primero estaba previsto ya que se alargó para que coincidiese el inicio de los Sprints con el inicio de semana. El último Sprint se alarga un par de días más de los previstos. A continuación se detalla el tiempo que se ha dedicado a cada Sprint:

- Sprint 1. 60 horas. 8 de junio hasta 20 de junio.
- Sprint 2. 40 horas. 20 de junio hasta 27 junio.
- Sprint 3. 40 horas. 27 junio hasta 4 de julio.
- Sprint 4. 40 horas. 4 de julio hasta 11 de julio.
- Sprint 5. 40 horas. 11 de julio hasta 18 de julio.

[5] `https://www.apachefriends.org/es/index.html`
[6] `https://httpd.apache.org/docs/current/howto/htaccess.html`

- Sprint 6. 40 horas. 18 de julio hasta 25 de julio.
- Sprint 7. 80 horas. 25 de julio hasta 5 de agosto.

A continuación, se detalla que ocurrió y que tareas se realizaron durante cada Sprint, se recuerda que las tareas están definidas en el cuadro 4.2:

5.3.1. Sprint 1

Durante este Sprint se realizaron las tareas TU01, TU02, TU03, TU04, TU05, TU08, TU09 y TU10.

Al inicio del Sprint se presentó el puesto de trabajo en Formación Lanzanet. Se instalaron los programas necesarios para trabajar,entre ellos *Xampp* para funcionar como servidor local y de base de datos y *Sublime Text* como editor de texto. Se decidió que la metodología que se seguiría fuese Scrum y se acordó hacer las reuniones de Sprint en cada inicio de semana.

En primer lugar se instaló Prestashop en el servidor local y se explicó como funcionaba tanto el *frontend* y el *backend*. Después se explicó como funcionaba internamente Prestashop para poder modificar los ficheros necesarios y se realizo la siguiente tarea: Hacer que no aparezca las direcciones, como se explica en la sección 5.1.1.

Una vez configurado Prestashop, se comenzó a instalar Moodle y surgió un pequeño problema durante la instalación, había que actualizar la base de datos a una versión superior. Se actualizó con un comando y se continuó la instalación. Y al terminar la instalación surgió otro problema, las páginas no cargaban porque estaban en un bucle infinito. Se buscó, otra vez, una solución en los foros y se encontró que había que borrar la carpeta de la caché.

5.3.2. Sprint 2

Durante este Sprint se realizaron las tareas TU06 y TU14. Se comenzó a realizar la tarea TU07, pero no se terminó.

En este Sprint se buscó información sobre sincronizar la cuenta de usuario de Prestashop y la cuenta de usuario de Moodle, que se explica detalladamente en la sección 5.1.3. Tras varios días buscando información, se encontró que en Moodle existe una opción para utilizar los datos de otra base de datos para poder hacer la identificación de las cuentas de usuario, lo cual se adapta a lo que se estaba buscando. Al realizar pruebas con una cuenta de usuario creado en Prestashop no conseguía identificarse en Moodle. El problema estaba en las codificaciones que utilizaba Moodle no coincidía con las de Prestashop, y no se consiguió encontrar más información sobre la codificación que utiliza Prestashop para sus contraseñas.

Casi al final del Sprint, se vio que se había avanzado poco con la sincronización de las cuentas y se decidió realizar otra tarea. Se añadieron los *plugins* que tenían relación con las redes sociales, como se explica en la sección 5.1.5. Se instaló en primer lugar el *plugin* de Facebook

y se configuró para que aparezca la página de Formación Lanzanet en la página principal de Prestashop y los comentarios que aparecen en los cursos. A continuación se instaló el *plugin* de Twitter para que aparezcan los últimos tweets del Twitter de Formación Lanzanet.

5.3.3. Sprint 3

Durante este Sprint se realizó parte de la tarea TU12.

En este Sprint se realizó la tarea que en el momento un alumno compra un curso en la parte de Prestashop se matricule en el curso en Moodle, como se explica en la sección 5.1.6. Como no se tiene aún una conexión entre las cuentas de usuario de Prestashop y de Moodle, se creó en cada plataforma una cuenta de usuario para realizar pruebas con ellas. Al buscar información sobre las distintas formas de matriculación, se observó que se podía matricular a los alumnos de forma manual. Se utilizó este mismo método desde fuera de Moodle para poder matricular a un usuario pero sin éxito, ya que necesitaba de varios permisos que me impedían acceder a los métodos internos de los ficheros de Moodle.

Al buscar otro método para activar el curso, se observó que se podía añadir los datos del usuario y del curso en la tabla de las matriculaciones para que detecte que se ha matriculado en el curso. Ahora se necesitaba añadir estas líneas tras validar la compra de los cursos. Se empezó a crear el módulo ConectaMoodle para añadir un método a un *Hook* que se ejecute después de la validación de la compra. Al terminar la semana no se terminó el módulo, por lo que se pospuso al siguiente Sprint.

5.3.4. Sprint 4

Durante este Sprint se terminó la tarea TU12.

Antes de empezar a desarrollar el módulo ConectaMoodle, se comprobó los datos necesarios que se requería de otras tablas de la base de datos de Moodle para poder añadir una línea en las tablas de las matriculaciones, como se explica en el apartado 5.1.6. Todos los datos se podían buscar fácilmente uniendo varias tablas, pero se necesitaba obtener el identificador del curso en Moodle, se ha añadido una nueva columna en la tabla **ps_product** para indicar el id del curso en los productos de Prestashop. Se modificó el archivo */classes/Product.php* para que reconozca la nueva columna y para evitar que genere errores.

El método que se añade al *Hook* recibe como parámetro el id del pedido y a través de este valor se puede obtener la información necesaria a través de varias sentencias SQL. Con esto se consigue matricular un usuario en el curso que acaba de comprar. Pero después de realizar varias pruebas se observa que:

- Un usuario puede volver a comprar un curso que ya ha comprado y esto provoca que haya un duplicado de primary key y salte una excepción de base de datos.
- Un usuario que no está registrado en Moodle y realiza una compra provoca que el método no funcione correctamente.

- Al comprar varios cursos solamente se registra el primer curso.

El primer problema se consigne solucionar con lo que se explica en la sección 5.1.7, creando un método que compruebe que el usuario no haya comprado el curso. En el caso de que tal usuario se haya matriculado, sustituye el botón de 'Añadir al carrito' por 'Ya has comprado el curso', que por el momento no realiza ninguna acción pero en un futuro se planea que enlace con el curso correspondiente.

El segundo problema está por solucionar, por la falta de la sincronización de las cuentas de Moodle y de Prestashop. Como medida preventiva, se ha eliminado la posibilidad de realizar una compra si no estas identificado en Prestashop. Aun así, en el caso de que no exista tal usuario en Moodle, termina la ejecución del método sin modificar nada.

El tercer problema fue un error a la hora de desarrollar el método ya que no lo planteé para realizar compras de varios cursos, después de modificar el método del módulo se solucionó fácilmente.

5.3.5. Sprint 5

Durante este Sprint se desarrolla parte de la tarea TU07.

Al comienzo de este Sprint, el supervisor recomendó buscar información sobre Single Sign-On (SSO) ya que permitía unificar la identificación de las cuentas de usuario de Moodle y Prestashop.

En primer lugar se probó con OpenID, existía un módulo que realizaba la tarea que se deseaba, pero era de pago por lo que se descartó su uso. Se utilizó una de las librerías[7] de PHP que proporciona la página OpenID para crear nuestro módulo. Se hicieron varias pruebas sin éxito por lo que se buscó otra forma de identificar a los usuarios.

En segundo lugar se probó a utilizar la API de Google para que los usuarios se identifiquen con su cuenta de Google, a través de Google Sign-in [10]. Durante varios días se estuvo adaptando la plataforma Prestashop para utilizar Google Sign-in, pero a la hora de utilizar el servicio de Google no funcionaba.

Viendo que utilizar SSO no daba resultados se intentó solucionar el problema que había con la codificación de la contraseña que se vio en el Sprint 2. Analizando el código de Prestashop, se descubrió el método **encrypt()** donde se realizaban todas las codificaciones en Prestashop. Para realizar la codificación utiliza un valor que se guarda en las *cookies* junto al valor que le envían al método. Se modificó el método para recoja un atributo más que se utilizará como Salt para hacer la codificación. Aparecieron varios errores cuando se hicieron pruebas, había algunos archivos que no tenían modificado el método **encrypt()** pero al añadirle el atributo nuevo se solucionaron. Gracias a esto se podía identificar un usuario en Prestashop sin problemas y con una capa más de seguridad. Ahora falta modificar la codificación en Moodle para que pueda utilizar la misma cuenta de usuario tanto en Prestashop como en Moodle.

[7] `http://openid.net/developers/libraries/`

5.3.6. Sprint 6

Durante este Sprint se termina de desarrollar la tarea TU07 que quedo pendiente del Sprint anterior y se terminó la tarea TU06.

Antes de comenzar con las tareas pendientes del anterior Sprint se observó que las cuentas de usuario que se creaban no codificaban la contraseña como se había desarrollado en el anterior Sprint. Dentro de la clase *Customer*, que representa la clase modelo de la tabla de los clientes, existe un método llamado **setWsPasswd()** que se utiliza para guardar la contraseña. En este método se añadió código para que compruebe si la variable *secure_key* no tiene un valor asignado, se genere y guarde uno y si contiene algún valor, se utilice este valor. Una vez asegurado que la variable *secure_key* contiene un valor, se puede utilizar para codificar la contraseña. Gracias a estas modificaciones se soluciona el problema.

Una vez corregido el problema, se comenzó a modificar el fichero **/moodle/auth/db/auth.php**. Dentro del cual contiene el método que identifica al usuario usando las tablas de la base de datos que se le indica en la configuración. El problema está en que la codificación que se utiliza en Prestashop es diferente a la que puede realizar Moodle. Por eso se añade la misma forma de codificar la contraseña usando las mismas variables que se usa en Prestashop. Para ello se recoge el valor de *secure_key* de la tabla *ps_customer* y se codifica junto a la contraseña que recoge del formulario, y se comprueba que coincide con el valor de *passwd* guardado en la tabla *ps_customer* de Prestashop.

Al hacer pruebas de identificación en Moodle con una cuenta de Prestashop no funcionó. Se descubre un error en el fichero de identificación de Moodle, en el bucle *foreach* generándose un excepción *OutOfBound* y se modifican varias líneas de código.

La lista *selectfields* contiene los nombres de los campos que se indican en la configuración de Moodle a la hora de indicar la base de datos externa y la lista *fields* contiene los valores que se recoge de la tabla de Prestashop.

Antes:

```
$fields = array_values($fields);
foreach (array_keys($selectfields) as $index => $localname) {
    $value = $fields[$index];
```

Después:

```
//$fields = array_values($fields);
foreach ($selectfields as $index => $localname) {
  $value = $fields[$localname];
```

Realizando estos cambios se soluciona el problema y se consigue, por fin, identificarnos en Moodle con la misma cuenta de Prestashop. Hasta el final del Sprint se buscó información sobre realizar una identificación externa a Moodle para poder identificarnos en ambas plataformas al mismo tiempo, se encontró en la página web StackOverflow una pregunta con información que puede servir [11], pero se adaptará en el siguiente Sprint.

5.3.7. Sprint Final

Durante este Sprint se continua trabajando en la tarea TU07.

Este Sprint se decidió terminar la tarea de sincronizar la identificación de los usuarios en ambas plataformas aplicando la solución que se propone en [11]. Pero para poder aplicar esta solución es necesario instalar una librería de PHP llamada pecl_http[8]. Tras la instalación se generaron varios errores. Por lo que esta solución se descarta por la falta de tiempo y con ello la tarea TU07 no se completa.

A continuación se realiza la tarea de modificar el formulario de edición de un producto de Prestashop para poder seleccionar la identificación de un curso de Moodle, como se explica en 5.1.6. Se crea un nuevo método que se añade al *Hook* DisplayAdminProductsExtra en el que se recoge la lista de los nombres de los cursos en Moodle y se envían junto a un fichero con la plantilla que se quiere añadir. Junto a este método se crean otro método que se añadirá al *Hook* ActionProductUpdate y servirá para guardar los datos que se envían desde el formulario.

Tres días antes de que se termine el Sprint se inicia el traspaso del proyecto de un servidor local a uno externo. Se utiliza el programa Filezilla [9] para subir los ficheros por vía FTP al servidor. Los detalles de cómo se puso en marcha se explican en la sección 5.2.

Una vez terminado el traspaso de servidor, se hicieron varias pruebas para comprobar que funcionase correctamente. Se descubrió que en Prestashop al añadir cosas al carrito de la compra no avanzaba a la confirmación del pedido. Por lo visto el error ocurría por el método **encrypt()** que se modificó en el Sprint 5. Los ficheros que se han modificado para arreglar los problemas que surgieron al modificar el método vuelven a una versión anterior sin modificar y únicamente se modifica el fichero *model/Customer.php*, como se explica en 5.1.4.

5.4. Tareas sin finalizar

A pesar de que se previó el tiempo necesario para cada tarea, han habido algunas tareas que no se han podido realizar. Las tareas que están pendientes son:

- TU07 - Sincronizar la identificación de las cuentas en ambos lados.
- TU11 - Proporcionar y configurar una pasarela de pago.
- TU13 - Proporcionar SEO.

Estas tareas no se han podido realizar porque han fallado las previsiones del tiempo necesario para realizar las tareas, que se hicieron en el cuadro 4.2. Aunque han habido tareas que su tiempo ha sido menor, otras en cambio ha aumentando considerablemente el tiempo que se ha dedicado a desarrollarse. En la siguiente tabla se realiza una comparativa del tiempo previsto, del tiempo real y de la diferencia entre ambos:

[8] `http://pecl.php.net/package/pecl_http`
[9] `https://filezilla-project.org/`

Id. Tarea	Puntos	Tiempo (horas)	Tiempo Real (horas)	Diferencia
TU01	1	4	4	0
TU02	6	32	32	0
TU03	0	0	0	0
TU04	0	0	0	0
TU05	1	4	0	-4
TU06	6	24	96	+72
TU07	20	80	128	+48
TU08	1	4	4	0
TU09	0	0	0	0
TU10	0	0	0	0
TU11	8	32	—	—
TU12	10	40	64	+24
TU13	4	16	—	—
TU14	4	16	4	-12

Cuadro 5.1: Comparación del tiempo en cada tarea.

Se recuerda que cada punto de historia de usuario equivalía a 4 horas de trabajo. El tiempo que se indica en la columna Tiempo Real, es una aproximación a las horas realizadas ya que no se cronometró la duración de las tareas.

Como se puede observar en el cuadro 5.1, hay tres tareas que tienen un tiempo invertido mucho mayor que el que se calculó en primer lugar:

- TU06 - Sincronizar cuentas de usuario en la base de datos. Uno de los principales problemas que se encontró al realizar esta tarea es que no se tuvo en cuenta que Prestashop y Moodle utilizaban codificaciones diferentes para sus contraseñas. Se dedicó mucho tiempo a la exploración del funcionamiento interno de Prestashop y a acceder a la variable que se guardaba en la cookie de Prestashop en Moodle sin éxito. Y después, cuando se modificó el método de codificación en Prestashop, se generaban errores a medida que se iba navegando por la plataforma, por lo que también se dedicó tiempo a solucionar estos problemas hasta que se optó por la solución final.

- TU07 - Sincronizar la identificación de las cuentas en ambos lados. Se ha dedicado más tiempo a esta tarea que a otras tareas, y aun así no se ha encontrado una solución que funcionase. Se han probado diferentes soluciones a lo largo de los Sprints y ninguna ha podido realizar la tarea deseada, ya sea por problemas de configuración o bien que no se ha podido instalar varios elementos necesarios para su uso. Al final se no se terminó la tarea ya que se vio que faltaba poco tiempo y aun habían tareas pendientes que hacer.

- TU12 - Matricular al alumno en el curso adquirido. Uno de los problemas al realizar esta tarea fue la modificación de la tabla de *ps_customer* y la modificación de la clase *Customer*. Además no se conocía en profundidad el funcionamiento de los *Hooks* ni cual de todos los disponibles era el correcto para el uso que quería darse.

Si se hubiese conocido un poco mejor el funcionamiento interno de las plataformas y se hubiese planificado correctamente la duración de cada tarea, posiblemente se hubiesen completado a tiempo las tareas pendientes.

Capítulo 6

Conclusiones

A lo largo de mi estancia en las prácticas he adquirido conocimientos sobre el funcionamiento de dos de las plataformas web que son conocidas mundialmente y he podido experimentar con ellas para unificarlas. He podido aplicar lo aprendido durante todos mis años de carrera. He aprendido a utilizar un lenguaje nuevo de programación, PHP, que es bastante solicitado por las empresas tecnológicas hoy en día. Además, he podido conocer de primera mano lo que es estar en un ambiente de trabajo.

Durante mi estancia en las prácticas, se aplicó Scrum que es una de las metodologías ágiles que aprendimos en la universidad, aunque con unas pequeñas variaciones ya que durante todo el proyecto estábamos el supervisor y yo, y me ha servido para aprender a aplicar una metodología en el ámbito empresarial. Gracias a esta metodología se conocía en todo momento las tareas pendientes y las realizadas. Y si se añadía alguna variación o una tarea nueva se podía planificar con facilidad.

El tema del proyecto me pareció interesante de desarrollar porque conocía Moodle de usarlo como estudiante de la UJI y quería conocer como funcionaba por dentro. También me ha sorprendido Prestashop, ya que hasta entonces había leído alguna noticia en blogs de Internet pero nunca había trabajado con esta plataforma, y cuando lo instalé me sorprendió que después de la instalación tengamos un e-commerce funcional. Aunque me ha resultado un poco tedioso ver el funcionamiento interno de cada una de las plataformas, se agradece enormemente que el código de ambas plataformas estuviese comentado. Aun así, hecho en falta una guía en la documentación de cada plataforma que indique a grandes rasgos el funcionamiento de sus plataformas.

Para finalizar, mi experiencia durante las prácticas ha sido muy positiva y enriquecedora para mi futuro como ingeniero informático.

Bibliografía

[1] Documentación de Prestashop. http://doc.prestashop.com/ [Consulta: 8 de junio de 2016]

[2] Moodle. https://moodle.org/?lang=es [Consulta: 8 de junio de 2016]

[3] Mike Cohn. Succeeding with agile: software development using Scrum. Addison-Wesley, 2010.

[4] Documentación de Prestashop. http://doc.prestashop.com/display/PS15/Hooks+in+PrestaShop+1.5 [Consulta: 4 de julio de 2016]

[5] Prestashop Forum. https://www.prestashop.com/forums/topic/458303-remove-delivery-and-invoice-address-fields-in-the-guest-checkout-ps-16/ [Consulta: 14 de junio de 2016]

[6] Rengochan blog. http://www.regochan.com/moodle/item/83-sincronizar-usuarios-y-matriculaciones-en-moodle-desde-bases-de-datos-externas [Consulta: 20 de junio de 2016]

[7] crackstation.net. https://crackstation.net/hashing-security.htm [Consulta: 14 de julio de 2016]

[8] Twitter Widgets https://twitter.com/settings/widgets

[9] Pablohunny blog. http://www.pablohunny.co.uk/2013/03/failing-at-understanding-moodle.html [Consulta: 5 de julio de 2016]

[10] Google Sign-In https://developers.google.com/+/web/signin/

[11] Stack Overflow. http://stackoverflow.com/questions/16017713/moodle-accept-login-from-external-site [Consulta: 20 de julio de 2016]